UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/673,587	09/29/2003	Jonathan Appavoo	YOR920030317US1 (16856)	1607

23389 7590 04/17/2008

SCULLY SCOTT MURPHY & PRESSER, PC
400 GARDEN CITY PLAZA
SUITE 300
GARDEN CITY, NY 11530

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
04/17/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/673,587	APPAVOO ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 March 2008 (claims submitted 2/4/08)*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 3/20/08, observing the previously submitted claims as per 2/4/08 (which were not entered then as per the Advisory action of 3/6/08), which are herein entered as per the current submission.

As indicated in Applicant's response, claims 6, 9, 14, 17, 20, 24 have been amended. Claims 1-24 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).
A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 6, 14, 21 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 5, 14 of copending Application No. 11/227,761 (hereinafter '761).

Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations. Following are but a few examples as to how the certain claims from the instant invention and from the above copending application are conflicting with each other.

As per instant claim 6, '761 claim 5 also recites dynamically update an operating system via hot swapping a new object for its old instance object; and although '761 claim 5 does not recite replacing while the operating system remains active to provide continual availability of resources to applications in the computer system, this 'hot swapping' is deemed obvious variant language for dynamic operation of an OS *while said operating system remains active ... provides continual availability of hardware resources by applications in the computer system.* Further, '761 claim 5 recites an obvious variant of claim 6' language in terms of changing reference pointer from the old object to the new object; that is, this reference pointer would be equivalent to *identifying references* to said first code component and *replacing the identified references to said first code component to said new code component* . Moreover, '761 claim 5 does not explicitly recite separating the first code component into objects grouped in table, whereby identified references to said objects are arranged and entered in said table. But in view of '761 claim 6 reciting *an underlying data structure* used by the old object instance to point from the updated object instance to this underlying data structure, a reference table is suggested for the dynamic correspondence between new and old object instance would point all references from one old object to the other new object to effectuate a hot replacement or swapping. Thus, it would have been obvious for one skilled in the art to provide said runtime factory object of '761 in form of underlying mapping table whose entries are arranged for storing reference pointers (as suggested in '761 claim 5) because these entries can act as references that would be pointing mutual correspondence between old and new object instance during the dynamic replacement of reference objects as recited in the instant claim 6, in view of well-known approach using

reference table to interrelate dynamic code referencing to provide program references resolution support at runtime as suggested in '761 hot swapping from above.

As per instant claims 14, and 21, these claims recite the main limitations of instant claim 1, hence would also have been obvious variations of '761 claim 5 in light of the above analysis.

As per claims 6, 14, and 21, '761 claim 14 also recites updating of an *operating system* without rebooting, identifying references (i.e. *determining ... old class definition meets ... requirement of a new class definition; structures indicate ... instantiations of the old class definition* -- Note: structure representing definition of old or new class reads on references of first or new code component; i.e. said structure being identified for said determining leading to the hot-swapping) to said first code component and replacing the identified references to said first code component to said new code component (i.e. *hot swapping each ... object instance for its corresponding old object instance*) and *underlying data structure* to support old object to be pointed to the new object (see '761 claim 14). '761 claim 18 does not explicitly recite separating the first code component into objects grouped in table, whereby identified references to said objects are arranged and entered in said table; but in view of the underlying structure to store the references pointing to *object instance* being swapped from above, this reference *table* (for grouping entries arranged to act as referencing old and new object instance in terms of mutual correspondence) limitation would have been obvious in light of the rationale as set forth above.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 10-11, 13-17 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

The current focus of the Patent Office in regard to statutory inventions under 35 U.S.C. § 101 for method claims and claims that recite a judicial exception (software) is that the claimed invention recite a practical application. Practical application can be provided by a physical transformation or a useful, concrete and tangible result. The following link on the World Wide Web is the United States Patent And Trademark Office (USPTO) reference in terms of guidelines on a proper analysis on 35 U.S.C. §101 rejection.

<http://www.uspto.gov/web/offices/pac/dapp/opla/preognotice/guidelines101_20051026.pdf>

Specifically, claim 17 recites a system for swapping source code in an OS system, comprising *means for* identifying and for replacing and executing the replacing. The Disclosure teaches using C++ language as means to implement the mechanism for hot swapping based on Clustered Object and using set of tables (see Specifications, pg. 5-6) as well as Mediator Object (Specs, pg. 10). The recited means for identifying and replacing amount to functionality based on software-implemented OO code; thus the means as claimed are devoid of support by any hardware in order to carry out said software functionality. As set forth in the USC 101 Guidelines (Annex IV, pg. 52-54) the claimed means are merely listed 'functional descriptive

material' and as such cannot be materialize via hardware-based execution to yield a tangible, useful and concrete result. Besides, software listing per se does not constitute any of the 4 categories of statutory subject matter. The claim is rejected for not being a statutory category and for remaining a non-practical application.

Claims 11, 13-17 are rejected for failing to provide hardware support to embody the above means or to execute the software functionality thereof.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claim 17 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. This claim recites "A method according to claim 10" in line 1, whereas claim 10 is system claim. There is insufficient antecedent basis for this limitation in the claim.

8. Claim 10-17 are rejected for reciting 'swapping source code' in a computer system. This *source code* being nowhere disclosed in the Disclosure, the above limitation leads to the intepretation that some teachings is lacking in the context of the claim as whole. The claim recites swapping (first and new) code component, and one of ordinary skill would not be able to construe hot swapping of OS code components can be done so that 'source code' is replaced as recited. The above language amounts to what appears to be a gap of teaching or undefined relationship between the 'source code' and the replacing steps on the claim. Claims 10-17 are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the

necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: how component code being swapped in the running OS without interruption can be a replacing of 'source code'.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Slingwine et al., USPN: 6,219,690 (hereinafter Slingwine).

As per claim 1, Slingwine discloses in a computer system using an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first source code component, (e.g. data coherence … hardware requirement … cache memories -col. 5, lines 4-20; shared-memory - col. 6, lines 41-61), a method of replacing said first source code component with a new source code component, the method comprising:

identifying references to said first code component (e.g. current generation 108 – pg. 3 – Note: CALLBACK processing to make next generation to become current generation – see col. 6, lines 41-50 – whereby associated structured context information for a thread activity – col. 9, li. 45-to col 10, li. 17; col. 11, lines 29-35; Fig. 4 – along with generation-contained callback constructs – see Fig. 4; *handle table*, *table_ptrs*, Fig. 6 col. 17 line 52 to col. 18, line 18 -- are replaced **reads on** identifying references of first code component); and

replacing the identified references to said first source code with references to said new source code component (e.g. col. 5 lines 50-52; next generation 110, Fig. 3; UPDATES: Add to next generation 90 – Fig. 3; Fig. 4; col. 11 line 29-67).

Slingwine does not explicitly disclose replacing while said operating system remains active and while said operating system provides continual availability to applications of the hardware resources by applications operational in the computer system. However, Slingwine mentions about transparency to user in regard preventive and corrective actions, all of which being prohibitive because overhead or hardware implication (col. 4, lines 30-50), thus recommend a callback based approach to avoid deadlocks with minimum overhead resources (e.g. *no data locks, elimination of overheads ... repairing deadlocks* - col. 5, lines 57-63; see Fig. 2; *kernel running* - col. 10, li. 54-67), hence has suggested maintaining availability of resources to the user applications while effectuating replacement at a low level being inexpensive in resources usage at a overall level. It would have been obvious for one skill in the art at the time the invention was made to implement the low overhead call back-based approach by Slingwine so that availability to user level applications of resources would not be impeded by the low overhead and dynamic replacement as purported above, because according to Slingwine, data can be maintained coherently at minimum cost to user level and the need to repair memory conflicts via undue resources expenses (as in deadlocks) as set forth above could be avoided.

As per claim 2, Slingwine discloses low-level interrupt (e.g. Fig. 5A, 5B, 5C, 5D; *interrupt* -- col. 18, lines 42-55 – Note: interrupt periodically invoked without need for user input suggests mutual exclusion of threads via callback implementation being low and transparent to application layer – see *kernel 36, interrupt* under *user process* – Fig. 2) but does not explicitly

disclose method being transparent, however this transparency has been addressed as obvious from claim 1.

As per claim 3, Slingwine discloses wherein the method is scalable (e.g. *global generation, possible large number of processes* – see col. 18, lines 42-46; *expanded* – col. 17, lines 41-67; *per-thread where possible ... some other entity where possible* – col. 10, li. 44-47).

As per claim 4, Slingwine discloses a multiprocessor system with plurality of processors so that the method is implemented on each processor independently (e.g. *per-processor context* - Fig. 4; col. 19, li. 1-3).

As per claim 5, Slingwine discloses

establishing a quiescent state for the first code component (e.g. col. 7, lines 50-67);

transferring said identified references at said established quiescent state (e.g. *unlinking ... adds a callback to the next generation... links the new element ... erases the original element* – col. 8 lines 54-67; Fig. 3; col. 9, lines 18-44) from the first code component to the new code component; and

after transferring said identified references to the new code segment at said established quiescent state (e.g. *through a quiescent state ... allowing updater ... safely* - col. 10, lines 7-51; col. 8, lines 54-67), swapping the first code component with the new code component (e.g. col. 10, lines 7-51; col. 11 line 29-67 -- Note: context switching using callback transfer and quiescent state monitoring for switching from an old generation to a next generation by placing all list of callbacks into the next generation **reads on** swapping – see col. 11, lines 29-35; Fig. 4).

As per claim 6, Slingwine discloses separating the first code component into objects; and grouping said objects into a table (e.g. Fig. 6; *one bit per thread ... next level ... group of*

threads and so on – col. 9, lines 31-44; col. 12, lines 5-15 – Note: thread and associated data structures per threat activity reads on objects and references thereto being grouped into structure – see col. 12, lines 35-53 – or pointer table – see Fig. 6-- to support *Updaters* - col. 17 line 52 to col. 18, line 18), arranging said grouped references to identify said references to said objects, which references to said objects are entered in the structure or call back table (e.g. col. 12, lines 5-15, 35-53; *handle table*, *table_ptrs*, Fig. 6; col. 17 line 52 to col. 18, line 18).

As per claim 7, Slingwine discloses wherein the replacing step includes the steps of: establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component and identifying active threads as said references (refer to claim 5); transferring said identified references during the quiescent state from the first code component to the new code component; and after transferring said said identified references, swapping the first code component with the new code component (refer to claim 5).

As per claim 8, Slingwine discloses wherein the replacing step includes the steps of: establishing a quiescent state for the first code component that includes the identified references (refer to claim 7); transferring the identified references at the quiescent state from the first code component to the new code component by providing a infrastructure operating a best transfer algorithm (see Fig. 3-5 – Note: mutual exclusion policy via passing of changes as context structure updates to the next thread context reads on best algorithm infrastructure – refer to claim 6); and after transferring said quiescent state, swapping the first code component with the new code component (refer to claim 7); and further discloses transferring by providing an infrastructure to negotiate a best transfer algorithm (refer to claim 7 – Note: monitoring for a

quescient state to provide data update to thread structures reads on negotiating for a best algorithm).

As per claim 9, Slingwine discloses wherein the step of identifying references includes the steps of

separating the first code component into objects, and grouping said objects into a table, arranging said table to identify said references to said objects, which identified references are entered in the table (refer to claim 6); and

the replacing step includes the steps of

establishing a quiescent state for the first code component that includes the identified references; transferring the identified references at the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and after transferring said references to the new code, swapping the first code component with the new code component (refer to claims 5, 7, or 8).

As per claim 10, Slingwine discloses a system for swapping source code in a computer system including an operating system, said operating system including at least one source code component the system comprising:

means for identifying, while said operating system is active and providing continual access to said resources, references to a first source code component of the operating system; and

means for replacing the identified references a new code with transferring the identified references to the new code component; and executing the replacing;

all of which steps of identifying and replacing having been addressed in claim 1.

Slingwine does not explicitly disclose swapping while the OS providing continual availability to applications of hardware resources by applications operational in the computer system, and replacing while said operating system is active and providing continual access to said resources to said first source code with references to a new source code component for the operating system. However, the continual providing of resources to the user application while replacing is happening at a lower level has been rendered obvious in claim 1.

As per claim 11, refer to the rejection of claim 2-3.

As per claims 12-17, refer to claims 4-9, respectively.

As per claim 18, Slingwine discloses a program storage device, for use with a computer system including an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first source code component (Fig. 1-2), said program storage device being readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for replacing said first source code component with a new source code component, the method steps comprising:

identifying references to said first source code component; and

replacing the identified references to said first source code with references to said new source code component;

all of which being addressed in claim 1.

Slingwine does not explicitly disclose replacing while said OS remains active in providing continual availability to applications of hardware resources by applications operational in the computer system; but this limitation has been addressed in claim 1.

As per claims 19-24, refer to claims 4-9, respectively.

Response to Arguments

11. Applicant's arguments filed 3/20/08 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

Double Patenting Rejection:

(A) Applicants have submitted that instant claims 6, 14, 21 are not anticipated or obvious over '761 claims 1 or 18 in terms that if dependent claims are patentable over any section under § 102 or 103, the claims dependent thereto should be patentably distinct (under MPEF 608.01n (II) – see Appl. Rmrks pg. 11, middle. It is noted that the grounds of rejection in the obvious-type Double patenting has been modified to meet the current state of the submitted response. The rejection has addressed where the language specifics in both corresponding parts of '761 application and the instant application would be equivalent or variant of each other in terms of obviousness. The Applicants have not provided evidences as to prove how the mapping of construct as set forth in the rejection would be improper; instead, it is argued that the dependent claims are patentable because the independent claims are not anticipated by or obvious over any prior art. The argument is not persuasive to overcome obviousness of the type of rejection, which based on respective claim language distinction or conflicts with respect to one set of claims over another, not prior art.

USC § 102 Rejection:

(B) Applicants have submitted that the invention is about seamlessly swapping of component in an OS that continues to provide availability of resources during the swapping; when in fact Slingwine discloses concurrent reading and updating data for coherency maintaining (Appl. Rmrks pg. 12). The general remark does not apply any particular language of the claims so to

provide convincing evidence such as to demonstrating that a corresponding cited portion in Slingwine would fail to fulfill. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(C) Applicants have submitted that the invention is about hot swapping based on establishing a quiescent state and negotiated protocol to effectuate transferring of elements and rerouting of code components; and that Slingwine focus of coherency of data using callbacks to provide mutual exclusion of threads cannot be perceived as teaching the above hot swapping (Appl. Rmrks pg. 13-14). For each part of claims 1, 10, an d 18, the rejection has provided teaching that would be deemed fulfilling the language of the corresponding limitation. Applicants provide a alleged conclusion that Slingwine mutual exclusion is not same as hot swapping as contemplated by the invention, but this argument does not persuasively point to a specific construct of any claim and thereby establishing how the cited portions in the rejection fail to fulfill, render obvious. The rejection has addressed constructs in the claim each in its own context, and what appears to be the purport of the invention – as described in Applicant's Remark, pg. 13 - cannot be read into the claim. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(D) Applicants have submitted that Slingwine does not teach the mutual exclusion such that it is for replacing while the operating system continues to provide access to hardware resources

(Appl. Rmrks pg 14, bottom). The argument is now moot in light of the new grounds of rejection.

(E) Applicants have submitted that Slingwine' s use of callback processing to cause a next generation to become the current generation, and such in the context of thread activity (Appl. Rmrks, pg. 15 top); and this is not 'identifying references to the first code component'. It is noted that the above limitation is not provided with sufficient details in order to preclude the arranging of structure and pointer by Slingwine in order to, based on callback structure monitoring and quiescent events, make the switch of thread-related context (refer to Rejection - including transferring of related data) from fulfilling this 'identifying references' step as recited. The argument is deemed not convincing.

(F) Applicants have submitted that Slingwine's as cited in Fig. 3-4, does not teach 'replacing the identified references ... to said new code component' (Appl. Rmrks, pg. 15, middle). Code component as claimed is understood as component that is involved in providing access to hardware resources of the OS; and Slingwine's thread and association with their next generation of callback are treated as component supporting access of critical data in a shared memories, so that the references being identified are the very pointer tables or structure to associate the context played by the threads in order to provide timely switching of thread context based on a quiescent state, to avoid undue spending of corrective measures. The language of the claim has been deemed fulfilled by the respective mappings using Slingwine; and the above argument appears to be a general statement which does not effectively put forth any distinction between any particular claim phrase against any particular portion in Slingwine.

In all, the arguments are not persuasive, and the claims will be rejected as set forth in the Office Action.

Conclusion

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

April 11, 2008